Certificate Authority (private)

General Information

This CA assumes the following:

- Best practice (2015) regarding crypto, key-length etc.
- The Root-CA is provided by the (fictive) company «TheBC»
- We don't create an intermediate certivicate authority
- All commands assumes to be executed within `/opt/CA`
- CRL distribution point is at URI:http://crl.thebc.ch/TheBC-CA.crl.pem
- OCSP is provided at URI:http://ocsp.thebc.ch:8888

CheetSheet

Create Server Certificate

KEY: `# openssl genpkey -out private/<hostname>.key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:4096`
CSR: `# openssl req -config TheBC-CA.conf -key private/<hostname>.key.pem -new -out csr/<hostname>.csr.pem`
SIGN: `# openssl ca -config TheBC-CA.conf -extensions v3_OCSP -in csr/<hostname>.csr.pem -out certs/<hostname>.cert.pem`
CHECK: `# openssl x509 -noout -text -in certs/<hostname>.cert.pem`

Possible Server Types: `[v3_CA|v3_intermediate_ca|v3_OCSP|usr_cert|server_cert]`

Revoke Certificate

REVOKE: `# openssl ca -config TheBC-CA.conf -revoke certs/<cert-name>.cert.pem`
REBUILD CRL: `# openssl ca -config TheBC-CA.conf -gencrl -out crl/TheBC-CA.crl.pem`

Create the CA

Create Directory Structure

First we need to crate a home for our Certificate Authority

```
# mkdir -p /opt/CA/private /opt/CA/certs /opt/CA/crl /opt/CA/csr /opt/CA/newcerts
```

Then we need to create some file:

1. A random file which will be used by the CA
2. A index file to store the created certificates
3. A "index" for the CRL (Certificate Revocation List)
4. Change the permissions for the ./private directory

```
# openssl rand -out /opt/CA/private/RANDFILE 8192
# touch /opt/CA/index.txt
# echo 01 > crl/crl.number
# chmod 700 /opt/CA/private
```

Create CA Config

For our CA to operate and to be consistent we are specify the configuration in a global configuration file.

NOTE: Normaly this file is named openssl.cnf.

```
# vi /opt/CA/TheBC-CA.conf
```

```
####################################################################
# CA Definition `man ca`
#
# The [ ca ] section is mandatory. Here we tell OpenSSL to use the
# options from the [ CA_default ] section.
#
[ ca ]
default_ca      = CA_default                    # The default ca section

####################################################################
# The [ CA_default ] section contains a range of defaults.
# Make sure you declare the directory you chose earlier (/opt/CA).
[ CA_default ]

# Directory and file locations.
dir               = /opt/CA                                 # Where everything is kept
certs             = $dir/certs                  # default place for new certs
crl_dir           = $dir/crl                            # clr directory
new_certs_dir     = $dir/newcerts
database          = $dir/index.txt                      # database index file
serial            = $dir/serial                         # Don't initialize with echo, use -create_serial
RANDFILE          = $dir/private/RANDFILE       # private random number file

# The root key and root certificate
private_key       = $dir/private/TheBC-CA.key.pem       # CA private key
certificate       = $dir/certs/TheBC-CA.cert.pem                # CA certificate

# For certificate revocation lists
crlnumber         = $dir/crl/crl.number         # current crl number
crl               = $dir/crl/ca.crl.pem         # current crl
crl_extensions    = crl_ext                                 # [ crl_ext ]
default_crl_days  = 30                                      # The default life for a certificate and a CRL.

default_md        = sha512                                  # Default digest algorithm
name_opt          = ca_default                          # Subject Name options
cert_opt          = ca_default                          # Certificate field options
default_days      = 365                                     # The default life for a certificate and a CRL.
preserve          = no                                  # openssl will re-order the attributes in the DNs of CSRs
policy            = policy_anything                     # [ policy_anything] default policy section to use

####################################################################
# An alternative policy not referred to anywhere in this file. Can
# be used by specifying '-policy policy_anything' to ca(8).
#
[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

####################################################################
# This is where we define how to generate CSRs
# [ req ] section are applied when creating certificates or certificate signing requests.
# Options for the `req` tool (`man req`)
[ req ]
default_bits            = 4096                                          # Size of keys
distinguished_name      = req_distinguished_name        # [ req_distinguished_name ]
string_mask             = utf8only                                      # permitted characters
default_md              = sha512                                        # message digest algorithm

####################################################################
# Per "req" section, this is where we define DN info
# The [ req_distinguished_name ] section declares the information normally required in a certificate signing request.
You can optionally specify some defaults.
#
[ req_distinguished_name ]
#countryName                    = Country Name (2 letter code)
#countryName_default            = US
#countryName_min                = 2
```

```
#countryName_max            = 2
#stateOrProvinceName        = State or Province Name (full name)
#stateOrProvinceName_default = NEW YORK
#localityName               = Locality Name (city, district)
#localityName_default       = NEW YORK
0.organizationName              = Organization Name (company)
0.organizationName_default      = TheBC
organizationalUnitName          = Organizational Unit Name (department, division)
organizationalUnitName_default  = Certificate Authoriy Departement
emailAddress                    = Email Address
emailAddress_max                = 64
emailAddress_default            = sysop@thebc.ch
commonName                      = Common Name (hostname, IP, or your name)
commonName_max                  = 64

####################################################################
# The next few sections are extensions that can be applied when signing certificates.
# For example, passing the -extensions v3_ca command-line argument will apply the options set in [ v3_ca ].

####################################################################
# We'll apply the v3_ca extension when we create the root certificate.
[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

####################################################################
# We'll apply the v3_ca_intermediate extension when we create the intermediate certificate. pathlen:0 ensures that
there can be no further certificate authorities below the intermediate CA.
[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

####################################################################
# Stanza for OCSP servers
[ v3_OCSP ]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = critical, OCSPSigning

####################################################################
# We'll apply the usr_cert extension when signing client certificates, such as those used for remote user
authentication.
[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
#nsCertType = server                    # ssl server
#nsCertType = objsign                   # ssl object signing
#nsCertType = client, email             # client
nsCertType = client, email
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

####################################################################
# We'll apply the server_cert extension when signing server certificates, such as those used for web servers.
[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
#nsCertType = server                    # ssl server
#nsCertType = objsign                   # ssl object signing
#nsCertType = client, email             # client
nsCertType = server
subjectKeyIdentifier = hash
```

```
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
crlDistributionPoints = URI:http://crl.thebc.ch/TheBC-CA.crl.pem
authorityInfoAccess = OCSP;URI:http://ocsp.thebc.ch:8888

####################################################################
# The crl_ext extension is automatically applied when creating certificate revocation lists.
[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always
```

Create the root key

Create the root key (TheBC-CA.key.pem) and keep it absolutely secure. Anyone inpossession of the root key can issue trusted certificates. Encrypt the root keywith AES 256-bit encryption and a strong password.

Note: Use 4096 bits for all root and intermediate certificate authority keys. You'll still be able to sign server and client certificates of a shorterlength.

```
# openssl genpkey -aes256 -out private/TheBC-CA.key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:4096
# chmod 400 private/TheBC-CA.key.pem
```

Create the root certificate signing request

Use the root key (ca.key.pem) to create a root certificate (ca.cert.pem).Give the root certificate a long expiry date, such as twenty years. Once theroot certificate expires, all certificates signed by the CA become invalid.

```
# openssl req -config TheBC-CA.conf -key private/TheBC-CA.key.pem -new -extensions v3_ca -out csr/TheBC-CA.csr.pem
```

Note: As a common-name use something like: TheBC-Root-CA

Self-Sign the certificate request

```
# openssl ca -create_serial -config TheBC-CA.conf -selfsign -days 3650 -extensions v3_ca -in csr/TheBC-CA.csr.pem -out certs/TheBC-CA.cert.pem
# chmod 444 certs/TheBC-CA.cert.pem
```

Here we specify the -days 3650 because the default in the `TheBC-CA.conf` is 365.

Verify the root certificate

```
# openssl x509 -noout -text -in certs/TheBC-CA.cert.pem
```

Keep an eye on the following parameters:

- the Signature Algorithm used (`sha512WithRSAEncryption`)
- the dates of certificate Validity
- the Public-Key bit length (`4096 bit`)
- the Issuer, which is the entity that signed the certificate
- the Subject, which refers to the certificate itself
- the Basic Constraints should be `CA:TRUE`
- the Key Usage should be `Digital Signature, Certificate Sign. CRL Sign`

NOTE: The Issuer and Subject are identical as the certificate is self-signed.Note that all root certificates are self-signed.

Create the Chain

cat intermediate/certs/intermediate.cert.pem certs/ca.cert.pem > intermediate/certs/ca-chain.cert.pem
chmod 444 intermediate/certs/ca-chain.cert.pem

If you don't have a intermediate CA the just copy the public certificate form the root-ca

```
# cp certs/TheBC-CA.cert.pem certs/TheBC-CA-chain.cert.pem
```

```
# chmod 444 certs/TheBC-CA-chain.cert.pem
```

Final Check

Below you'll find the CA-directory-tree we have so far.

```
:/opt/CA> tree
.
├── certs
│   ├── TheBC-CA.cert.pem                          # Public Cert of the CA
│   └── TheBC-CA-chain.cert.pem                 # Certificate Chain
├── crl
│   └── crl.number                                      # Index of the CRL (atm. 01)
├── csr
│   └── TheBC-CA.csr.pem                          # CSR for the CA (can be deleted)
├── index.txt                                                  # DB with the issued certificates
├── index.txt.attr                                      # Contains some extra attributes ???
├── index.txt.old                                          # Backup file
├── newcerts
│   └── B15E2E0017A94F06.pem                  # The same as TheBC-CA.cert.pem
├── private
│   ├── RANDFILE                                           # Contains some random numbers
│   └── TheBC-CA.key.pem                          # Private key of the CA (SENSITIVE!)
├── serial
└── TheBC-CA.conf                                          # The configuration file for the CA

5 directories, 12 files
:/opt/CA>
```

Create Server Certificates

Create the server key

```
openssl genpkey -out private/thebc.ch.key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:4096
```

Create the Certificate signing request

```
openssl req -config TheBC-CA.conf -key private/thebc.ch.key.pem -new -out csr/thebc.ch.csr.pem
```

Sign the CSR

```
openssl ca -config TheBC-CA.conf -extensions server_cert -in csr/thebc.ch.csr.pem -keyfile private/
ca.key.pem -out certs/thebc.ch.cert.pem

openssl ca -config TheBC-CA.conf -extensions server_cert -in csr/thebc.ch.csr.pem -out certs/
thebc.ch.cert.pem
```

Example

```
:/opt/CA> openssl ca -config TheBC-CA.conf -extensions server_cert -in csr/thebc.ch.csr.pem -out certs/
thebc.ch.cert.pem
Using configuration from TheBC-CA.conf
Enter pass phrase for /opt/CA/private/TheBC-CA.key.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 12780703370455895815 (0xb15e2e0017a94f07)
        Validity
            Not Before: Jun 24 20:09:41 2015 GMT
            Not After : Jun 23 20:09:41 2016 GMT
        Subject:
            organizationName          = TheBC
            organizationalUnitName    = Certificate Authoriy Departement
            commonName                = thebc.ch
            emailAddress              = sysop@thebc.ch
        X509v3 extensions:
            X509v3 Basic Constraints:
```

```
                CA:FALSE
            Netscape Cert Type:
                SSL Server
            X509v3 Subject Key Identifier:
                91:73:EA:91:8D:D9:F2:9D:EA:51:A0:0C:61:21:35:ED:1C:16:66:0D
            X509v3 Authority Key Identifier:
                keyid:B1:9C:83:84:2C:3B:64:B6:A0:BE:C2:CD:DB:20:70:C6:75:32:70:AE
                DirName:/O=TheBC/OU=Certificate Authoriy Departement/CN=TheBC-Root-CA/
emailAddress=sysop@thebc.ch
                serial:B1:5E:2E:00:17:A9:4F:06

            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.thebc.ch/TheBC-CA.crl.pem

            Authority Information Access:
                OCSP - URI:http://ocsp.thebc.ch:8888

Certificate is to be certified until Jun 23 20:09:41 2016 GMT (365 days)
Sign the certificate? [y/n]:
```

Deploy

To deploy the certificates to a server (like nginx) you'll need the following files:

- TheBC-CA-chain.cert.pem
- thebc.ch.key.pem
- thebc.ch.cert.pem

Certificate revocation lists

A certificate revocation list (CRL) contains information about which certificates have been revoked. A CRL can be used by a client application, suchas a web browser, to check a server's authenticity. A CRL can also be used by a server application, such as Apache or OpenVPN, to deny access to clients that are no longer trusted.

A CRL is normally published at a publicly accessible location. Third-parties can fetch the CRLfrom this location to check whether any certificates they rely on have been revoked.

Note: Some applications vendors have deprecated CRLs and are instead using the Online Certificate Status Protocol (OCSP).

Prepare the configuration file

A certificate authority normally encodes the CRL location into certificates that it signs. Add crlDistributionPoints to the appropriate sections, which usually means the [server_cert] section.

```
[ server_cert ]
# ... snipped ...
crlDistributionPoints = URI:http://crl.thebc.ch/TheBC-CA.crl.pem
```

Create the CRL

```
# cd /root/ca
# echo 01 > crl/crl.number
# openssl ca -config TheBC-CA.conf -gencrl -out crl/TheBC-CA.crl.pem
```

Note: The `CRL OPTIONS` section of the `` `man ca` `` page contains more informationon how to create CRLs.

View the CRL

```
# openssl crl -in crl/ca.crl.pem -noout -text
```

No certificates have been revoked yet, so the output will state `No Revoked Certificates`.

The CRL is usually re-created at regular intervals. By default, the CRL expires after 30 days. This is controlled by the `default_crl_days` option in the `[ CA_default ]` section.

Revoke a certificate

REVOKE: `# openssl ca -config TheBC-CA.conf -revoke certs/bob@example.com.cert.pem`
REBUILD CRL: `# openssl ca -config TheBC-CA.conf -gencrl -out crl/ca.crl.pem`

Server-side use of the CRL

For client certificates, it's typically a server-side application (eg, Apache) that is doing the verification. This application needs to have local access to the CRL.
In Alice's case, she can add the `SSLCARevocationPath` directive to her Apache configuration and copy the CRL to her web server. The next time that Bob connects to the web server, Apache will check his client certificate against the CRL and deny access.
Similarly, OpenVPN has a `crl-verify` directive so that it can block clients that have had their certificates revoked.

Client-side use of the CRL

For server certificates, it's typically a client-side application (eg, web browser) that is doing the verification. This application needs to have remote access to the CRL.
When signing a certificate with an extension that includes `crlDistributionPoints`, the certificate will contain a reference to the location specified. The application can read this information and fetch the CRL.

Online Certificate Status Protocol (OCSP)

The Online Certificate Status Protocol (OCSP) was created as an alternative to certificate revocation lists (CRLs). Similar to CRLs, OCSP enables a requesting party (eg, web browser or webserver) to determine the revocation state of a certificate.
When a CA signs a certificate, they will typically include an OCSP server address (eg, http://ocsp.thebc.ch) in the certificate. This is similar in function to `crlDistributionPoints` used for CRLs.
As an example, when a web browser is presented with a server certificate, it will send a query to the OCSP server address specified in the certificate. At this address, an OCSP responder listens to queries and responds with the revocation status of the certificate.

Note: It's recommended to use OCSP instead where possible, though realistically you will tend to only need OCSP for website certificates. Some web browsers have deprecated or removed support for CRLs.

Prepare the configuration file

To use OCSP, the CA must encode the OCSP server location into the certificates that it signs. Use the `authorityInfoAccess` option in the appropriate sections, which in our case means the `[ server_cert ]` section.

```
[ server_cert ]
# ... snipped ...
authorityInfoAccess = OCSP;URI:http://ocsp.thebc.ch:8888
```

Create a new stanza

```
[ v3_OCSP ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = OCSPSigning
```

Create the OCSP pair

The OCSP responder requires a cryptographic pair for signing the response that it sends to the requesting party. The OCSP cryptographic pair must be signed by the same CA that signed the certificate being checked.

KEY: `# openssl genpkey -out private/ocsp.thebc.ch.key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:4096`
CSR: `# openssl req -config TheBC-CA.conf -key private/ocsp.thebc.ch.key.pem -new -out csr/ocsp.thebc.ch.csr.pem`
SIGN: `# openssl ca -config TheBC-CA.conf -extensions v3_OCSP -in csr/ocsp.thebc.ch.csr.pem -out certs/ocsp.thebc.ch.cert.pem`
CHECK: `# openssl x509 -noout -text -in certs/ocsp.thebc.ch.cert.pem`

This shows something like this:

```
    X509v3 Key Usage: critical
```

```
        Digital Signature
    X509v3 Extended Key Usage: critical
        OCSP Signing
```

Revoke a certificate

The OpenSSL ocsp tool can act as an OCSP responder, but it's only intended for testing. Production ready OCSP responders exist, but those are beyond the scope of this guide.

1. Create a server certificate to test.
2. Run the OCSP responder on `localhost`. Rather than storing revocation statusin a separate CRL file, the OCSP responder reads `index.txt` directly. Theresponse is signed with the OCSP cryptographic pair (using the `-rkey` and `-rsigner` options).

```
# openssl ocsp -port 2560 -text -sha256 \
      -index index.txt \
      -CA certs/ca-chain.cert.pem \
      -rkey private/ocsp.thebc.ch.key.pem \
      -rsigner certs/ocsp.thebc.ch.cert.pem \
      -nrequest 1
```

In another terminal, send a query to the OCSP responder. The `-cert` option specifies the certificate to query.

```
# openssl ocsp -CAfile intermediate/certs/ca-chain.cert.pem \
      -url http://localhost:2560 -resp_text \
      -issuer certs/intermediate.cert.pem \
      -cert certs/test.example.com.cert.pem
```

The start of the output shows:
• whether a successful response was received (`OCSP Response Status`)
• the identity of the responder (`Responder Id`)
• the revocation status of the certificate (`Cert Status`)

```
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: ... CN = ocsp.example.com
    Produced At: Apr 11 12:59:51 2015 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: E35979B6D0A973EBE8AEDED75D8C27D67D2A0334
      Issuer Key Hash: 69E8EC547F252360E5B6E77261F1D4B921D445E9
      Serial Number: 1003
    Cert Status: good
    This Update: Apr 11 12:59:51 2015 GMT
```

Revoke the certificate.

```
openssl ca -config TheBC-CA.conf -revoke certs/<CERT>.cert.pem
```

As before, run the OCSP responder and on another terminal send a query. Thistime, the output shows `Cert Status: revoked` and a `Revocation Time`.

```
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: ... CN = ocsp.example.com
    Produced At: Apr 11 13:03:00 2015 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: E35979B6D0A973EBE8AEDED75D8C27D67D2A0334
      Issuer Key Hash: 69E8EC547F252360E5B6E77261F1D4B921D445E9
      Serial Number: 1003
    Cert Status: revoked
    Revocation Time: Apr 11 13:01:09 2015 GMT
    This Update: Apr 11 13:03:00 2015 GMT
```

TODO: OCSP Responder behind ngix

```
<VirtualHost x.x.x.x:80>
ServerName ocsp.cacert.org
DocumentRoot /dev/null
RewriteEngine on
RewriteCond %{CONTENT_TYPE} !^application/ocsp-request$
RewriteRule ^/(.*) http://localhost:2560/ [P]
CustomLog /var/log/apache/ocsp.cacert.org-access.log combined
ErrorLog /var/log/apache/ocsp.cacert.org-error.log
</VirtualHost>
```

References

- https://jamielinux.com/docs/openssl-certificate-authority/index.html
- http://www.phildev.net/ssl/
- https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices_1.2.pdf
- https://calomel.org/nginx.html
- http://nginx.org/en/docs/http/ngx_http_ssl_module.html#ssl_stapling_responder
- http://itigloo.com/security/generate-an-openssl-certificate-request-with-sha-256-signature/
- https://www.openssl.org/docs/apps/
- http://isrlabs.net/wordpress/?p=169